## REMARKS

Claims 1-9, 11-27, 29-43, 45-60 and 62-68 are pending. Reconsideration is respectfully requested in light of the following remarks.

**Section 103(a) Rejections:**

The Examiner rejected claims 1-8, 11-13, 16, 34-42, 45-47, 50, 52-59, 62-64 and 67 under 35 U.S.C. § 103(a) as being unpatentable over Monday et al. (U.S. Patent 6,263,377) (hereinafter "Monday") in view of Venners ("Inside the Java Virtual Machine"), and claims 9, 14, 15, 17-27, 29-33, 43, 48, 49, 51, 60, 65, 66 and 68 as being unpatentable over Monday in view of Venners and further in view of Babaoglu et al. ("Anthill: A Framework for the Development of Agent-Based Peer-to-Peer Systems") (hereinafter "Babaoglu"). Applicants respectfully traverse these rejections for at least the following reasons.

First, Applicants note that the present Office Action essentially reiterates the same points as before, and that the Examiner failed to substantially address numerous ones of the arguments previously presented in this case. For example, in response to all the arguments presented below save those directed towards avoiding class conflicts, the Examiner merely makes the following immaterial assertion:

> it is noted that the features upon which applicant relies (i.e., the remote class loader is not a custom class loader) are not recited in the rejected claim(s)…In addition, the remote class loader in the system of Monday performs its functions without communication with the default class loader, clearly, the remote class loader operate separately from and transparently to the default class loader. [*Sic*]

Applicants respectfully submit that comments regarding a custom class loader were provided to distinguish Applicants' current claimed limitations from the prior art (those described in Monday, Venners, and Applicants' own background section). As already noted, the custom class loaders are instances of subclasses of the Classloader, and **by definition**, and as well know by any one of ordinary skill in the art of class loaders,

cannot operate separately from and transparently to the default classloader and cannot be independent from the default classloader (recited in the claims). Furthermore, Applicants have already addressed the Examiner's unsupported conclusory statement that the remote class loader of Monday "clearly" operates separately from and transparently to the default class loader. Additionally, the Examiner does not address arguments regarding the Examiner's assertions of inherency, which are clearly unsupported by the prior art and violate the guidelines set forth in the MPEP. Applicants respectfully submit that the Examiner fails to address these arguments and that Applicants' statements regarding custom class loaders were provided as indications of the prior art, over which Applicants' current claim limitations clearly distinguish. Thus, the Examiner has yet to substantially address Applicants' arguments or provide a *prima facie* case of obviousness for the present claims. Applicants address further responses below.

**Contrary to the Examiner's assertion, Monday in view of Venners fails to teach or suggest a remote class loader mechanism configured to: detect the indication that the class is not loaded; obtain the class from a remote system via a network; and store the class in a location indicated by the class path of the default class loader on the system; wherein the remote class loader mechanism is configured to perform said detect, said obtain, and said store separate from and transparent to the default class loader**. In claim 1 of the present application, *detecting the indication that the class is not loaded,* said *obtaining the class from a remote system via a network,* and said *storing the class in a location on the system indicated by the class path of the default class loader means* are all performed separately from and transparently to the default class loader for the virtual machine. The default class loader attempts to load a class from a class path of the default class loader means. If the class is not located, the recited remote class loader detects the indication that the class needed to execute the code on the system is not stored in the one or more locations indicated by the class path, obtains the class from a remote system via a network, and stores the class in a location on the system indicated by the class path of the default class loader means. These operations are separate from and transparent to the default class loader. The default class loader

may then load the class from the location on the system indicated by the class path. There is no indication of this transparency in the cited references.

In col. 3, lines 38-56 and elsewhere, Monday states "If x.class is NOT located, then a **subclass of** the CLASSLOADER, a REMOTECLASSLOADER...". **By definition, since Monday's REMOTECLASSLOADER is a subclass of the CLASSLOADER, it cannot and does not operate separately from and transparently to the default CLASSLOADER.** Clearly, Monday does not anticipate claim 1 of the instant application.

Applicants note that the Examiner asserts, with respect to this limitation, "inherent from the remote class loader check the remoteclasspath, obtain the class and store it in the directory without consulting from the class loader; col. 2, lines 44-56". **The Examiner's assertion is entirely unsupported by the actual teachings of the reference**. Instead, in the cited art the remoteclassloader is called by the classloader in the event that the class is not found in the classpath. Applicant respectfully submits that it is clearly not inherent that Monday's remoteclassloader operates both separately from and transparently with respect to the classloader. As the Examiner is certainly aware, according to M.P.E.P. 2131.01 III, in regard to a theory of inherency "**evidence** must make clear that the missing descriptive matter is **necessarily** present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill." The Examiner has not provided anything but the Examiner's own unsupported speculation that the functionality relied upon by the Examiner is inherent in Monday. The rejection is not supported by the actual evidence of record.

**Furthermore, Monday in view of Venners fails to teach or suggest wherein the default class loader is independent from the remote class loader mechanism**. As already indicated above, the remoteclassloader (which the Examiner currently relies on as the claimed remote class loader) of Monday is a subclass of classloader. Thus, **by definition,** the Monday's remoteclassloader is not independent from the default class loader.

In regards to Venners, page 1, line 10-page 3, line 5 of the Background section of the instant application generally describes the dynamic loading of classes using class loaders in the prior art. In particular, page 3, lines 4-5 describes that: In Java, to use a class, the class has to be in the class path of the default class loader, or alternatively a **custom** class loader may be provided. The Venners reference is an overview of the Java Virtual Machine (JVM). Chapters 1-4 "give a broad overview of Java's architecture". A careful review of the Venners reference, particularly an overview given in Chapter 3, page 2, shows that Venners' description of Java's method of loading classes using class loaders is consistent with what is described in the Background section of the instant application. For example, Venners, in Chapter 3, page 2, gives the following description of how Java's class loading mechanism works:

> Imagine that during the course of running the Java application, a request is made of your [custom] class loader to load a class named Volcano. Your [custom] class loader would first ask its parent, the **class path class loader**, to find and load the class. The **class path class loader**, in turn, would make the same request of its parent, the installed extensions class loader. This class loader, would also first delegate the request to its parent, the bootstrap class loader. Assuming that class Volcano is not a part of the Java API, an installed extension, or **on the class path**, all of these class loaders would return without supplying a loaded class named Volcano. When the **class path class loader** responds that neither it nor any of its parents can load the class [i.e., the requested class is **not on a class path**], your [custom] class loader could then attempt to load the Volcano class in its **custom** manner, by downloading it across the network. Assuming your [custom] class loader was able to download class Volcano, that Volcano class could then play a role in the application's future course of execution.

In contrast to the above description from Venners, claim 1 of the instant application recites that a *default class loader for a virtual machine* determines that a class needed to execute code on the system is not stored in one or more local locations indicated by the class path (and thus fails to load the class). An indication is generated that the class is not loaded. A remote class loader mechanism then, transparently to the default class loader of the virtual machine, detects the indication, obtains the class from a remote system via a network, and stores the class in a location indicated by the class path. The default class loader (**not** a custom class loader, as in the Venners reference and the

Monday reference, and as the Java class loading mechanism described by Venners and described in the Background section of the instant application operates) then loads the class from the location indicated by the class path.

To summarize, Venners discloses that, if the "default class loader" (**class path class loader**) cannot locate a class in a location on its class path, the "default class loader" notifies a "custom class loader" associated with the class, which then attempts to load the class, possibly from a remote location. Similarly, the Monday reference describes, in col. 3, lines 38-56, a REMOTECLASSLOADER that clearly operates as a conventional custom class loader as is disclosed in Venners and in the background section of the present application. In contrast, claim 1 of the instant application recites that, if the "default class loader" cannot locate a class in a location on its class path, a "remote class loader mechanism", transparently to the default class loader, determines that the class was not located on the class path of the default class loader, locates the class on a remote system, and stores the class in a location indicated by the default class loader's class path. The default class loader then loads the class from the location. Thus, what claim 1 of the instant application recites is clearly distinct from the Java class loading mechanism as described by the Venners reference and from Monday's disclosed system.

As indicated above, in response to these arguments, the Examiner merely asserts that the present claims do not recite that the remote class loader is not a custom class loader and provides a conclusory statement regarding the transparent and separate operation of the remote class loader. **The Examiner apparently has failed to understand Applicants' argument.** The Examiner ignores the substance of Applicants' arguments regarding the limitation "wherein the default class loader is independent from the remote class loader mechanism" and does not provide any actual support for the conclusion that the remote class loader of Monday acts transparently to and separate from the default class loader. Applicants assert that the provision of conclusory statements does not provide a *prima facie* case of obviousness.

Applicants remind the Examiner that, to establish a *prima facie* case of obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. As shown above, the cited art does not teach or suggest all limitations of claim 1.

**Furthermore, the Examiner's stated reason for combining the references, "because Venners teaches the details how a Java application load and utilize class that are needed at runtime", is clearly not a valid reason for combining the references.** It is if anything simply a "motive" for understanding how the Monday system already works, since the Monday system already employs the conventional class loading method described in Venners. Furthermore, the present application describes the conventional method for dynamically loading classes in virtual machines such as JVMs as is described in Venners, and clearly discloses a system and method that is distinctly different than the conventional method described in Venners. Claim 1 of the instant application clearly recites elements that enable the distinctly different method described in Applicants' specification. Moreover, the Examiner's stated reason is merely conclusory. Applicants note that the Examiner's response to this argument simply reiterates the same reason and further asserts it would have been obvious "for understanding how class loader work" [*Sic*]. Applicants respectfully submit that understanding the operation of a class loader is clearly not a reason to combine references in the specific manner suggested by the Examiner.

**Furthermore, even if the Monday and Venners references were combined, the combination would not produce anything like what is recited in claim 1 of the present application.** It would simply produce a system that handles the loading of classes using the Java class loading method as disclosed in Venners, which it is clear that Monday's system already uses. In other words, it would simply produce the Monday system as disclosed in the Monday reference. **Since the Monday reference already employs the class loading mechanism disclosed in Venners, there is and can be no motivation to combine the references in any case.**

Thus, for at least the reasons presented above, the rejection of claim 1 is not supported by the cited prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 1 also apply to claims 34, 35, and 52.

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

## CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-65900/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: August 25, 2008